TP1 - Projeto de Execução Dinâmica de Processos

Sistemas Operacionais

O presente trabalho tem por objetivo explorar temas referentes ao escalonamento e troca entre processos que utilizam um dado processador. É previsto o desenvolvimento de um ambiente que empregue uma política de escalonamento específica, bem como gerencie a inclusão e remoção de processos que ocupam o processador. A carga de processos deverá ser realizada a partir de programas que utilizarão uma linguagem assembly hipotética.

Descrição de programas e características de execução e ocupação

O usuário deverá ser capaz de descrever pequenos programas a serem executados pelo ambiente. O ambiente de execução é baseado em acumulador. Assim, para a execução de um programa, dois registradores estão presentes: ***(i)*** o acumulador (acc) onde as operações são realizadas e ***(ii)*** o ponteiro da instrução em execução (pc). A linguagem de programação hipotética a ser empregada pelo usuário para a programação e que manipula os dois registradores descritos é apresentada na Tabela 1. Ali são definidos em quatro categorias, conforme listado na primeira coluna.

Tabela 1 - Mnemônicos e funções associadas

Categoria	Mnemônico	Função
Aritmético	ADD op1	acc=acc+(op1)
	SUB op1	acc=acc–(op1)
	MULT op1	acc=acc*(op1)
	DIV op1	acc=acc/(op1)
Memória	LOAD op1	acc=(op1)
	STORE op1	(op1)=acc
Salto	BRANY label	pc ← label
	BRPOS label	Se acc > 0 então pc <- op1
	BRZERO label	Se acc = 0 então pc <- op1
	BRNEG label	Se acc < 0 então pc <- op1
Sistema	SYSCALL index	Chamada de sistema

As instruções da categoria aritmético podem operar em modo de endereçamento direto ou imediato com a memória. No modo de endereçamento imediato, o valor de *op1* pode ser o valor real a ser considerado na operação. No Modo de endereçamento diretor, *op1* representa a variável na área de dados cujo valor deve ser capturado. A diferenciação entre os dois modos no código assembly a ser gerado deve ser dado pela presença do caractere sustenido (#) em frente ao operador *op1*, a fim de representar o modo imediato, ou a ausência deste caractere, a fim de representar o modo direto. Um exemplo desta situação é ilustrado na linha 3 no código de exemplo da Figura 1.

Duas instruções são utilizadas na categoria memória, e representam a leitura ou a escrita de dados em memória de dados. Assim como assumido para as instruções aritméticas, a instrução de leitura da memória de dados pode ser realizada tanto em modo imediato quanto direto, e a interpretação segue tal como descrito anteriormente. Um exemplo pode ser visto na Figura 1, na linha 2. Já o comando de escrita (i.e. STORE) dá suporte apenas ao modo direto de operação, ou seja, um dado somente pode ser escrito em uma variável declarada na área de dados.

Para os mnemônicos referentes a saltos condicionais, label representa a posição da área de código que deve ser assumida por PC. Há quatro tipos de saltos, sendo um incondicional e três condicionais, assumindo-se então as condições destacadas na Tabela 1. Para a criação de *labels* utilize a representação de um "nome" alfanumérico, seguido de dois pontos (:), conforme ilustrado na Figura 1, na linha 3. Um exemplo de uso de uma instrução de salto condicional é apresentado na mesma figura, na linha 5.

A última categoria da tabela 1 representa a instrução de operação com o sistema. Deverá ser possível a associação de 3 tipos de pedidos de operação com o sistema, representados pelos valores numéricos '0', '1' e '2'. Uma chamada de sistema referenciando o valor '0' (zero) caracteriza um pedido de finalização/encerramento do programa, tal como um *halt*. Uma chamada de sistema referenciando o valor '1' (um) caracteriza um pedido de impressão de um valor inteiro na tela. Uma chama de sistema referenciando o valor '2' (dois) caracteriza um pedido de leitura de um valor inteiro, que deverá ser feito via teclado. Às chamadas de sistema cujos valores '1' e '2' forem atribuídos, deverá ser associada uma situação de bloqueio deste processo, com um valor aleatório entre 1 e 3 unidades de tempo.

```
1  .code                 #Define o início da área de código
2    LOAD variable       # Carrega em acc o conteúdo de variable
3    ponto1: SUB #1      # Subtrai do acc um valor constante (i.e. 1)
4    SYSCALL 1           # Imprime na tela o conteúdo do acumulador
5    BRPOS ponto1        # Caso acc>0 deve voltar à linha marcada por "ponto1"
6    SYSCALL 0           # Sinaliza o fim do programa
7  .endcode              #Define o final da área de código

8  .data                 #Define o início da área de dados
9    Variable 3          # Conteúdo da posição 1 da área de dados é 3
10 .enddata              #Define o final da área de dados
```

Figura 1 - Código exemplo.

Como finalização da descrição dos processos, devem ser assumidas as seguintes características.

- Ocupação de memória
  - Cada instrução ocupa uma posição da memória, independente de sua categoria;
  - Cada variável ocupa uma posição da memória;
  - O número de posições de um programa é então definido pelo número de instruções somado ao número de variáveis;
  - A organização de ocupação da memória por parte dos processos está fora do escopo deste trabalho;
- Os valores atribuídos às variáveis ou às instruções que operam de modo imediato podem assumir valores tanto positivos quanto negativos
- Cada instrução executa em uma unidade de tempo

Política de escalonamento

Como política de escalonamento entre tarefas, deve ser implementada a política EDF (Earliest Deadline First). O EDF define um escalonamento baseado em prioridades: a escala é produzida em tempo de execução por um escalonador preemptivo dirigido a prioridades. É um esquema de prioridades dinâmicas.

As premissas que determinam o modelo de tarefas no EDF são:

a. As tarefas são periódicas e independentes
b. O deadline de cada tarefa coincide com o seu período ($D_i=P_i$)
c. O tempo de computação (C_i) de cada tarefa é conhecido e constante (Worst Case Computation Time), consistindo no tempo necessário para a execução completa da tarefa
d. O tempo de chaveamento entre tarefas (troca de contexto) é assumido como nulo.

A política de escalonamento no EDF corresponde a uma atribuição dinâmica de prioridades que define a ordenação das tarefas segundo os seus deadlines absolutos (d_i). A tarefa mais prioritária é a que tem o deadline d_i mais próximo do tempo atual. A cada chegada de tarefa, a fila de prontos é reordenada, considerando a nova distribuição de prioridades. A cada ativação de uma tarefa T_i, seguindo o modelo de tarefas periódicas, um novo valor de deadline absoluto é determinado. Para fins deste trabalho, assume-se que o conjunto de tarefas fornecido é escalonável.

Uma tarefa somente deixa o estado de ***running*** quando: ***(i)*** a tarefa tem seu ***tempo de computação*** alcançado (fim da execução no período P_i); ***(ii)*** existe uma tarefa de mais alta prioridade pronta para ser executada no período vigente; ou ***(iii)*** realiza uma chamada de sistema. Nos dois primeiros casos, a tarefa volta para a lista de prontos, e sua nova prioridade é calculada conforme especificado no EDF. No último caso, caso o pedido seja de uma impressão ou leitura a partir da chamada do sistema, esta tarefa deverá assumir o estado de bloqueado e um intervalo de tempo de permanência neste estado deverá ser assumido (aleatório entre 1 e 3 unidades de tempo). Passado este tempo, a tarefa pode avançar para o estado de ***pronto*** e sua nova prioridade é calculada conforme especificado no EDF.

Para o algoritmo de escalonamento implementado, deve-se assumir que, supondo que uma tarefa A venha a ser removida do processador para que uma tarefa B o ocupe, quando da retomada da tarefa A, esta deve seguir sua execução do último ponto de parada. O tempo necessário para troca de contexto deve ser desconsiderado para o presente trabalho.

Interface da aplicação

O ambiente a ser desenvolvido deve permitir definir qual(is) tarefa(s) será (serão) carregada(s), o(s) instante(s) de carga (*arrival time*), o tempo de computação de cada tarefa (C_i) e o período de cada tarefa (P_i), lembrando que neste trabalho o deadline de cada tarefa coincide com o seu período. Como resultado, o sistema deve mostrar de forma clara e inequívoca o escalonamento da(s) tarefa(s) ao longo do tempo, além de sinalizar a perda de deadline de uma tarefa, informando o nome da tarefa e o instante de tempo.

Informações adicionais

O trabalho deverá ser realizado em grupos de 4 alunos (obrigatoriamente). Deverá ser entregue o código fonte do programa desenvolvido, bem como um manual do usuário em formato PDF contendo as explicações de como compilar e executar o programa. A linguagem de programação utilizada para desenvolver o trabalho é de escolha do grupo, desde que seja possível compilar e executar o programa no ambiente computacional disponível nas salas de aula laboratório do prédio 32.

A data de entrega está prevista para o dia 21/09/2023, até às 17hs15min. As apresentações serão realizadas pelos alunos a partir do material postado no Moodle.

Cada aluno deverá assinalar no Moodle o grupo ao qual pertence e **um integrante** do grupo deverá escolher **um horário** para o grupo apresentar o trabalho no dia 21/09/2023 ou no dia 26/09/2023, dentre os horários disponíveis no Moodle.

O trabalho deverá ser entregue no moodle a partir de um arquivo compactado (.zip ou .rar). O nome do arquivo deverá ser tal que contenha o nome e sobrenome de todos integrantes do grupo. O material postado no moodle é de inteira responsabilidade do aluno. A presença de arquivos corrompidos, que impeçam a avaliação do trabalho pelo professor será considerada como a não entrega do trabalho. Também não serão considerados trabalhos com erro de compilação. Casos onde sejam identificados plágio/cópia, receberão nota zero.